|  |
| --- |
| SG Holidays |

***Resources and References***

**Android ImageView Example**

<http://www.mkyong.com/android/android-imageview-example/>

**ImageView Reference** <http://developer.android.com/reference/android/widget/ImageView.html>

**Using lists in Android** <http://www.vogella.com/tutorials/AndroidListView/article.html>

**Android ListView Example** <http://www.mkyong.com/android/android-listview-example/>

Table of Contents

[My Modules 1](#_Toc481751913)

[Section A: Display image using ImageView 1](#_Toc481751914)

[Section B: Implementing Complex ListView 3](#_Toc481751915)

[Handling the Problem Statement 9](#_Toc481751916)

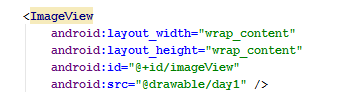
## Section A: Display image using ImageView

1. ImageView lets us display image easily. We are going to learn how to display images we already have (You may also find out more on how to get the image from the Internet when the app runs from the reference link above).
2. Create a new project and name it **MyImageView**.

|  |  |
| --- | --- |
| Project Name | Demo ImageView |
| Package | com.myapplicationdev.android.demoimageview |
| Activity Name | MainActivity |
| Layout Name | activity\_main.xml |
| Min SDK | API 16 |

\*Import into Github

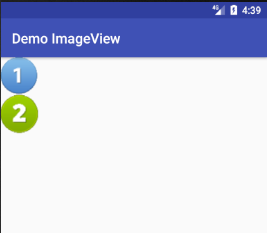
1. Switch to [Project View](https://myrp.sharepoint.com/sites/AndroidProgramming/AY2018-C347/_layouts/OneNote.aspx?id=%2Fsites%2FAndroidProgramming%2FAY2018-C347%2FSiteAssets%2FAY2018-C347%20Notebook&wd=target%28Android%20Studio.one%7C096D2DCC-B59B-4D66-BF6F-7E8A1E7F1A27%2FUser%20Guide%7C4ECAF804-DAF9-4E1A-9440-ECB63DE67722%2F%29) in the Project Explorer (Refer to OneNote, User Guide). Drag and drop the 5 images from LEO (**day1.png, day2.png**…**day5.png**) into **app > src > main > res > drawable** resource folder. By default, the files will be moved into the project.
2. Next, delete the automatically created “Hello World!” TextView, change your layout to **LinearLayout (Vertical)** and drag and drop an **ImageView** object onto your layout file.
3. The IDE will prompt you to set a default image for the ImageView. It will automatically generate the line android:src like the sample given below. In some cases, you’ll see app:srcCompat instead if you have the SupportPackage turned on.



The method above shows the way to set a default picture for the ImageView declaratively.

1. You may also set the image programmatically in your code. Drag another **ImageView** onto your layout (leave the ID as **@+id/imageView2**). You can delete the android:src or app:srcCompat line after choosing any image. The ImageView will show nothing in Design View of the layout file.
2. Add the Java code below to set the image to **day2** and run the app to see if you can see the image? Do you realise we use **R.drawable.day2** as the resource for the file named **day2.png**? That is, we use the filename without the extension .png.



1. You should observe the following outcome.   
     
   
2. Continue to display all day 1 to day 5 on the Activity (ImageView displaying image declaratively or dynamically). Commit and push the project into Github.

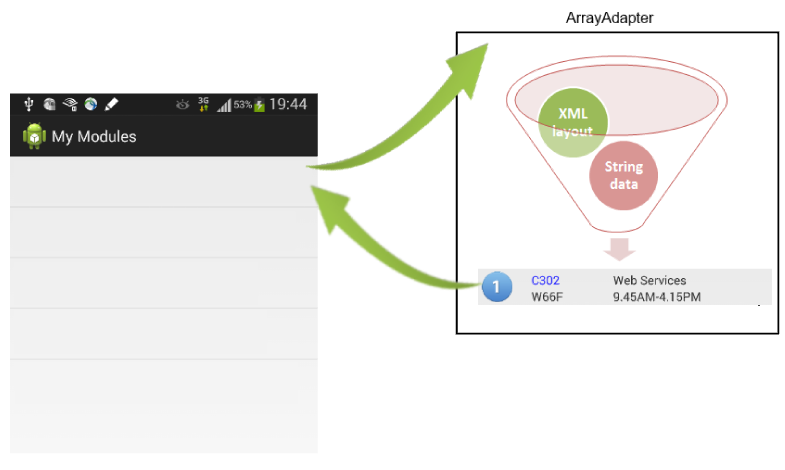
## Section B: Implementing Complex ListView

*<Reading section>*

ListView is a View that shows items neatly row by row. It is a very clean and tidy way of showing a long list of information.

The steps to implement a ListView are summarized below (this is only summary for your knowledge, you would only need to do the steps in point 3).

* 1. Just like any typical application, create an **Activity** class. This will be the main “screen” that hosts the ListView.
  2. Drag a ListView object into the layout and assign an appropriate id to it. (Eg **@+id/lvItems** instead of a generic @+id/listview)
  3. Create an *XML row layout file* to define how a row of data will be presented.
  4. Create a new Adapter for the ListView by creating a class and make it extends **ArrayAdapter**. The ListView object would ask this adapter every time it needs to display a row of data and this adapter will have to:
     1. Get the data for that row,
     2. Present the data in a View object with the look and feel defined by the row layout file in point c above and
     3. Return the View object to the ListView.
  5. Link the Activity in point a, the XML row layout file in point c and ArrayAdapter in point d together by setting the adapter to the ListView.



1. Now let’s first create a new project and name it **Demo ListView**. Leave the activity java filename as **MainActivity.java** and layout file as **activity\_main.xml**.

|  |  |
| --- | --- |
| Project Name | Demo ListView |
| Package | com.myapplicationdev.android.demolistview |
| Activity Name | MainActivity |
| Layout Name | activity\_main.xml |
| Min SDK | API 16 |

\*Import into Github

1. We are looking at displaying the name of some food and whether it deserves a star or not. We would create a Java class named **Food** to represent the data.
2. What are the fields and their data types to represent name of food and star (whether the food deserve a star)? List them below:
3. To give you a head start, this is one suggested template for the Food class.

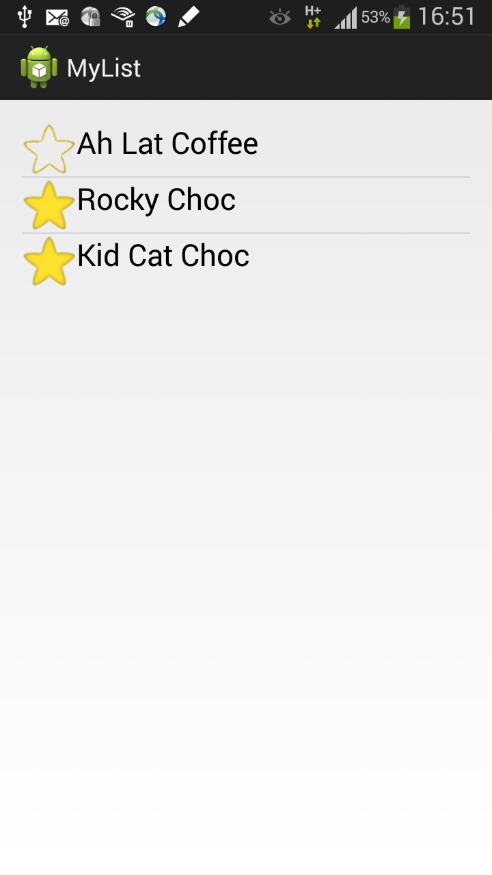
|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | public class Food {  private String name;  private boolean star;  public Food(String name, boolean star) {  this.name = name;  this.star = star;  }  public String getName() {  return name;  }  public boolean isStar() {  return star;  }  } |

1. Change the root layout in **activity\_main.xml** to **LinearLayout (Vertical)**. Drag a **ListView** object (under Composite) to be under the root layout. Change the ID to **@+id/lvFood**.
2. Next we would add a new layout file and name it **row.xml** by clicking the menu **File > New > XML > XML Layout File**. Change its root layout to **LinearLayout**. This layout defines how a row in ListView would look like. Do set its orientation to horizontal.
3. Drag **star.png** and **nostar.png** (download from LEO) into **drawable** folder. If you are having trouble putting the files in, look [**here**](#Drawables).
4. Drag an **ImageView** object onto **row.xml** and leave the image unassigned. Change its ID to **@+id/ivStar**.
5. Drag a **TextView** object onto **row.xml** and change its ID to **@+id/tvFoodName**.
6. Next create a new class and name it as **FoodAdapter**. Make this class extend **ArrayAdapter<Food>**. ArrayAdapter<Food> holds Food objects.
7. Modify the FoodAdapter class to the following code. Try to understand what the code is doing by studying the comments.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | public class FoodAdapter extends ArrayAdapter<Food>{  private ArrayList<Food> food;  private Context context;  private TextView tvFoodName;  private ImageView ivStar;    public FoodAdapter(Context context, int resource, ArrayList<Food> objects){  super(context, resource, objects);  // Store the food that is passed to this adapter  food = objects;  // Store Context object as we would need to use it later  this.context = context;  }  // getView() is the method ListView will call to get the  // View object every time ListView needs a row  @Override  public View getView(int position, View convertView, ViewGroup parent) {  // The usual way to get the LayoutInflater object to  // "inflate" the XML file into a View object  LayoutInflater inflater = (LayoutInflater) context  .getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  // "Inflate" the row.xml as the layout for the View object  View rowView = inflater.inflate(R.layout.*row*, parent, false);  // Get the TextView object  tvFoodName = (TextView) rowView.findViewById(R.id.*tvFoodName*);  // Get the ImageView object  ivStar = (ImageView) rowView.findViewById(R.id.*ivStar*);    // The parameter "position" is the index of the  // row ListView is requesting.  // We get back the food at the same index.  Food currentFood = food.get(position);  // Set the TextView to show the food    tvFoodName.setText(currentFood.getName());  // Set the image to star or nostar accordingly  if(currentFood.isStar()) {  ivStar.setImageResource(R.drawable.*star*);  }  else {  ivStar.setImageResource(R.drawable.*nostar*);  }  // Return the nicely done up View to the ListView  return rowView;  }  } |

1. Last few steps are to create a few Food objects to populate the Food array and to call **setAdapter ()** to link everything together as explained in point 2e above.

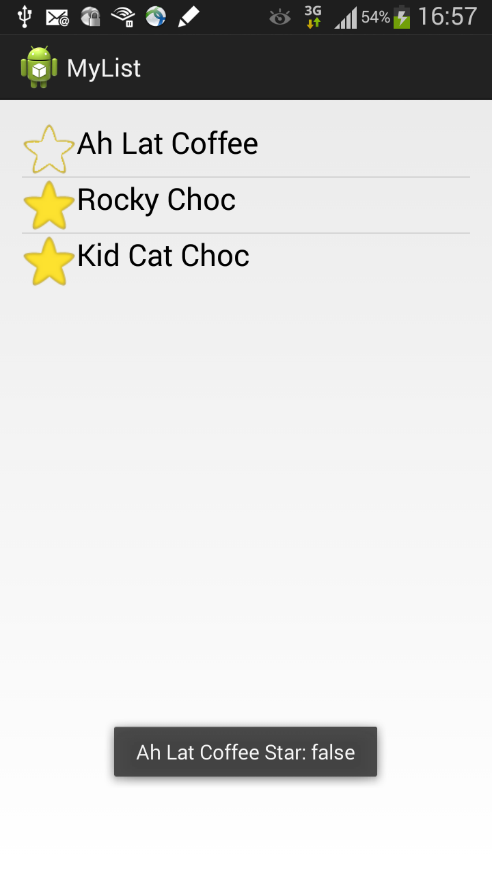
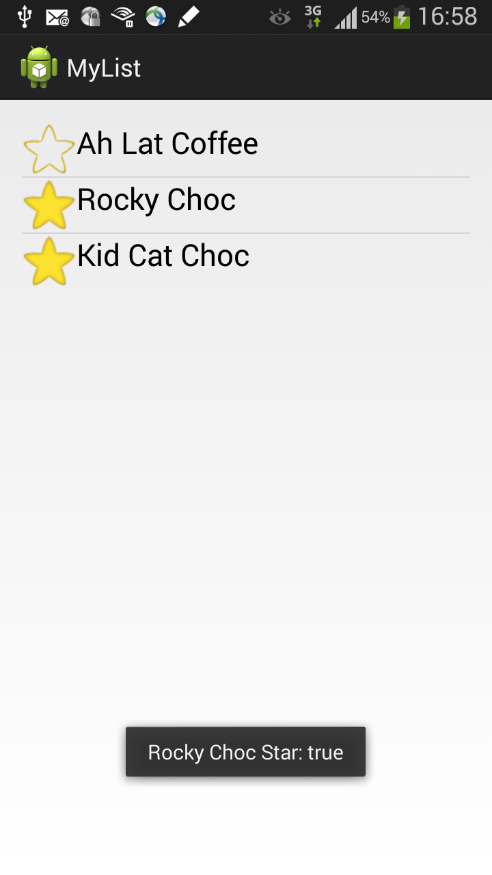
|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13  14  15  16  17  18  19  20  21  22  23  24  25  26 | public class MainActivity extends AppCompatActivity {    ListView lv;  ArrayAdapter aa;  ArrayList<Food> food;    @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  lv = (ListView) this.findViewById(R.id.*lvFood*);  // Create a few food objects in Food array  food = new ArrayList<Food>();  food.add(new Food("Ah Lat Coffee", false));  food.add(new Food("Rocky Choc", true));  food.add(new Food("Kid Cat Choc", true));  // Link this Activity object, the row.xml layout for  // each row and the food String array together  aa = new FoodAdapter(this, R.layout.*row*, food);  lv.setAdapter(aa);  }  ...  } |

1. Run the app and do you see a similar app as below?  
     
   

Commit and push the project into Github.

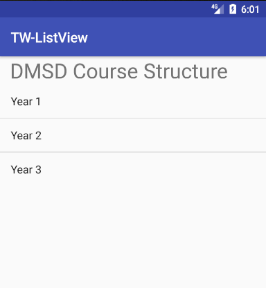
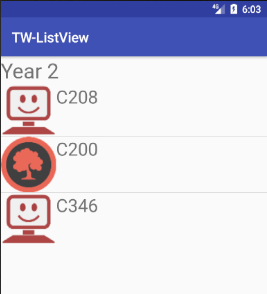
1. To handle the event when user clicks on an item, add the OnItemClickListener below to the ListView in MainActivity class.

|  |  |
| --- | --- |
| 1 2 3 4 5 … 13  14  15  16  17  18  19  20  21  22  23  24 | public class MainActivity extends AppCompatActivity {  …  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  …    lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {    @Override  public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  Food selectedFood = food.get(position);  Toast.*makeText*(MainActivity.this, selectedFood.getName()  + " Star: " + selectedFood.isStar(),  Toast.*LENGTH\_LONG*).show();  }  });  }  } |

1. You should be able to get the Toast to show when you click on the items as below  
     
    
2. Would you be able to make app start another Activity when an item is clicked? (Hint: refer to P01 on how to start another Activity). Commit and push the project into Github.

**Working together using Github**

Below is an Android App containing 2 Activities. The first Activity contains a simple list view while the SecondActivity contains a custom ListView.

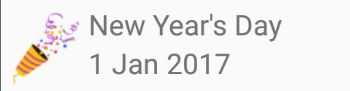
FirstActivity SecondActivity

1. You’ll get to practice how to create an app like shown above. Check out the skeleton project from <https://github.com/jason-lim-rp/TW-ListView>
2. With another teammate, do the following
   1. One of the teammate should import the project from repo (read [here](https://myrp.sharepoint.com/sites/AndroidProgramming/AY2018-C347/_layouts/OneNote.aspx?id=%2Fsites%2FAndroidProgramming%2FAY2018-C347%2FSiteAssets%2FAY2018-C347%20Notebook&wd=target%28GitHub.one%7CB0A54599-688F-4005-A883-8FC177D9396A%2FImport%20a%20project%20from%20Github%7C9A58FFB3-0850-43BF-9148-DE747DF85AAC%2F%29))
   2. Share the project into one of the teammate’s repository (read [here](https://myrp.sharepoint.com/sites/AndroidProgramming/AY2018-C347/_layouts/OneNote.aspx?id=%2Fsites%2FAndroidProgramming%2FAY2018-C347%2FSiteAssets%2FAY2018-C347%20Notebook&wd=target%28GitHub.one%7CB0A54599-688F-4005-A883-8FC177D9396A%2FImport%20into%20Github%20%28Setting%20up%20repo%5C%29%7C30CA541C-DF64-4BB6-BF65-859A5D2AB258%2F%29))
   3. Add the other teammate as contributor to the repository (read [here](https://myrp.sharepoint.com/sites/AndroidProgramming/AY2018-C347/_layouts/OneNote.aspx?id=%2Fsites%2FAndroidProgramming%2FAY2018-C347%2FSiteAssets%2FAY2018-C347%20Notebook&wd=target%28GitHub.one%7CB0A54599-688F-4005-A883-8FC177D9396A%2FAdding%20additional%20contributors%20to%20a%20repository%7CA6643167-AF45-4284-9904-5E071D651E71%2F%29))
   4. Complete the app where each teammate work on different part of the project. Commit and push the progress into Github regularly. (read [here](https://myrp.sharepoint.com/sites/AndroidProgramming/AY2018-C347/_layouts/OneNote.aspx?id=%2Fsites%2FAndroidProgramming%2FAY2018-C347%2FSiteAssets%2FAY2018-C347%20Notebook&wd=target%28GitHub.one%7CB0A54599-688F-4005-A883-8FC177D9396A%2FCommit%20changes%20to%20repository%7C00DB9B7E-5AA4-44FB-99D7-4319A7293995%2F%29))
3. If you face any issues with Github, you can refer the online guide in this module.
4. Write the name of the teammate you are working with and the repository here:

1. What are the missing parts in the projects? Compare and contrast it with [Section B](#_Section_B:_Implementing).

## Handling the Problem Statement

1. You would need to create an Activity that its layout contains a ListView object
2. You can store the holiday information in a class just like the Food class we had. What fields are necessary to capture the data to be displayed? Look at the problem statement again to find out. You could draw a class diagram here.
3. For the row XML layout file, what are the Views needed to construct it? You may need to use a few LinearLayout to achieve the neat UI.



1. Use ImageView to show the icon for the public holiday. How would you associate the holiday with an icon?
2. How many Activities do you need for the Problem? You may need to plan on the components for each Activity.